Claims

1. A method for debugging a software system, the software system having a first and second software component and a first coordinator for implementing a predetermined coordination protocol, each of the first and second components connected to the coordinator by a respective pair of complimentary coordination interfaces, the method comprising:

generating a record of software system events, each event record within the record of system events representing an inter-component control or dataflow interaction;

creating a behavioral template based on a predetermined behavior of the software system;

identifying an occurrence of the predetermined behavior within the record of software system events, based on the behavioral template.

2. A method according to claim 1 wherein creating the behavioral template comprises creating a visual prototype, which represents the predetermined behavior of the software system.

3. A method according to claim 1 wherein creating the behavioral template comprises creating a behavior expression, which represents the predetermined behavior of the software system.

4. A method according to claim 1 wherein generating the record of software system events comprises simulating an execution of the software system, with the record of software system events generated by the simulator.

5. A method according to claim 1 wherein generating the record of software system events comprises:

instrumenting the software system to provide an event notification to a runtime operating system for each software system event;

deploying the software system to a target architecture;

on the target architecture, capturing all notifications from the software system and storing the event notifications; thereby creating a record of software system events.

6. A method according to claim 1 wherein the predetermined behavior comprises a predetermined set of state changes selected from an execution of the software system.

7. A method according to claim 1 wherein the predetermined behavior comprises a predetermined set of state changes and message events selected from an execution of the software system.

8. A method for transforming a visual prototype into a behavioral expressions comprising:

creating an event causality graph for the visual prototype

removing any causally redundant edges in the event causality graph;

creating a cluster representing any concurrent nodes that progress forward together in time.

9. A method according to claim 8 wherein transforming a visual prototype into a behavioral expression further comprises:

creating a representation for each created cluster of nodes;

representing any causal chains between clusters in terms of a causal relation; and

branching each causal chain, if branching is needed in order to account for any overlapping clusters within the causal chain.

10. A method according to claim 8 wherein creating the event causality graph comprises:

creating an event node for an event record in the visual prototype; and adding an edge between a first and a second event, from the event record, to represent explicit causality between the first event and the second event.

11. A method according to claim 10 wherein creating the event causality graph further comprises adding an edge between a third and a fourth event , from the record of system events, representing implicit causality between the third and the fourth events based on the ordering of the third and the fourth event within a component trace.

12. A method according to claim 11 wherein removing the causally redundant edge comprises checking the immediate predecessor of each event to determine whether an event is causally related to its immediate predecessor.

13. A behavioral analysis method for use with a space/ time diagram comprising:

evaluating all system events from a record of system events one event at a time while maintaining causal relationships between system events; and

analyzing the evaluated system events in order to find a predetermined system behavior.

14. A behavioral analysis method according to claim 13 wherein evaluating system events one at a time comprises:

creating a topological sort of the system event records;

placing all system events in an explicit dependency graph in order to determine immediate precedence for each system event; and

assigning each system event a vector time to determine general causal relationships between events.

15. A behavioral analysis method according to claim 14 wherein the vector time is also used to determined concurrence between events.

16. A behavioral analysis method according to claim 15 further comprising replacing a found instance of the predetermined behavior with a replacement sequence of events, thereby reducing visual clutter on the space/ time diagram and highlighting the predetermined behavior.

17. A behavioral analysis method according to claim 16 wherein the replacement sequence of events is an abstract event of a higher level than the system events that comprise the predetermined system behavior.

18. A software system design tool comprising:

a simulator for simulating an execution of the software system;

a template tool for creating a behavioral template based on a predetermined behavior of the software system;

a debugging tool for identifying an instance of the predetermined behavior of the software system from a simulated execution of the software system based on the behavioral template.

19. A software system design tool according to claim 18 wherein the template tool allows a designer to create a behavioral template based on a visual prototype.

20. A software system design tool according to claim 18 wherein the template tool allows a designer to create a behavioral template based on a behavior expression.

21. A software system design tool according to claim 18 wherein the behavior of the software system comprises a predetermined set of software system events.

22. A software system design tool according to claim 18 wherein the behavior of the software system comprises a predetermined set of state changes.

23. A software system design tool according to claim 18 wherein the behavior of the software system comprises a predetermined set of state changes and system events.

24. A software design tool for use in a coordination-centric design environment comprising:

an automaton that analyzes system events and determines causal relationships between system events in order to identify a predetermined system behavior.

25. A software design tool according to claim 24 wherein the automaton analyzes system events one at a time in order to create a topological sort of the system events.

26. A software design tool according to claim 25 wherein the automaton places each system event in a dependency graph, thereby determining immediate precedence for each event.

27. A software design tool according to claim 26 wherein each event is given a vector time stamp, thereby determining general causal relationships and concurrence between events.

28. A software design tool according to claim 27 wherein the automaton identifies the predetermined system behavior based on the topological sort of system events, the dependency graph of events, and the vector timestamp of each event.

29. A software design tool according to claim 28 wherein the automaton replaces the identified system behavior with a replacement sequence of events.

30. A software design tool according to claim 29 wherein the replacement sequence of events are at a higher level of abstraction than the events within the identified system behavior.